January, 2003

Some Performance Analysis Issues

Adolfy Hoisie, Darren Kerbyson, Scott Pakin, Fabrizio Petrini, and Harvey Wasserman
hjw@lanl.gov
Modeling, Informatics, and Algorithms Group (CCS-3)
http://www.c3.lanl.gov/par_arch
Computer & Computational Sciences Division (CCS)

(Some Data from Mike Gittings, CRESTONE Team)

LAUR 03-0529

Los Alamos NATIONAL LABORATORY

Overview

- We were asked to comment on observed performance.
- Tactical goal is to examine efficiency and give an explanation.
- Strategic goal is to understand how applications match to architectures.
- Result: <u>A</u> view on the right questions to ask

Los Alamos NATIONAL LABORATORY

Approach

- Separate single-processor and multi-processor issues.

$$T_{run} = T_{computation} + T_{communication} \; (- \; T_{overlap})$$

$$T_{run} = f\,(T_{1\text{-}CPU}\;,\; \textit{Scalability function})$$

  - Single-processor performance issues addressed via measurements, especially hardware counters.
    - Not predictive but diagnostic
  - Multi-processor performance issues addressed via modeling.
    - Predictive but empirical.
- Assumptions:
  - Workload characterization (code module, etc.)
  - Focused on "compute-bound" portions (exclude OS, I/O)

Pitfalls

- Workload not representative
  - Profiling and creation of stand-alone representative benchmarks are vital.
  - "Benchmarking is the process by which we determine performance *on a workload of interest*."
    - Be careful generalizing results from one workload to another.
- Quoting single-processor performance figures from multi-processor measurements.

Single-Processor Performance

- Observed Rate = f * Peak_Rate

  f ≡ 0.333333 sPPM* Only

  (~~1500~~ 3000 FLOPS per gridpoint)

  * 3-D gas dynamics via simplified Piecewise Parabolic Method

- f = ~0.1 ± .05 Everything else

  (≤ 50 FLOPS per gridpoint)

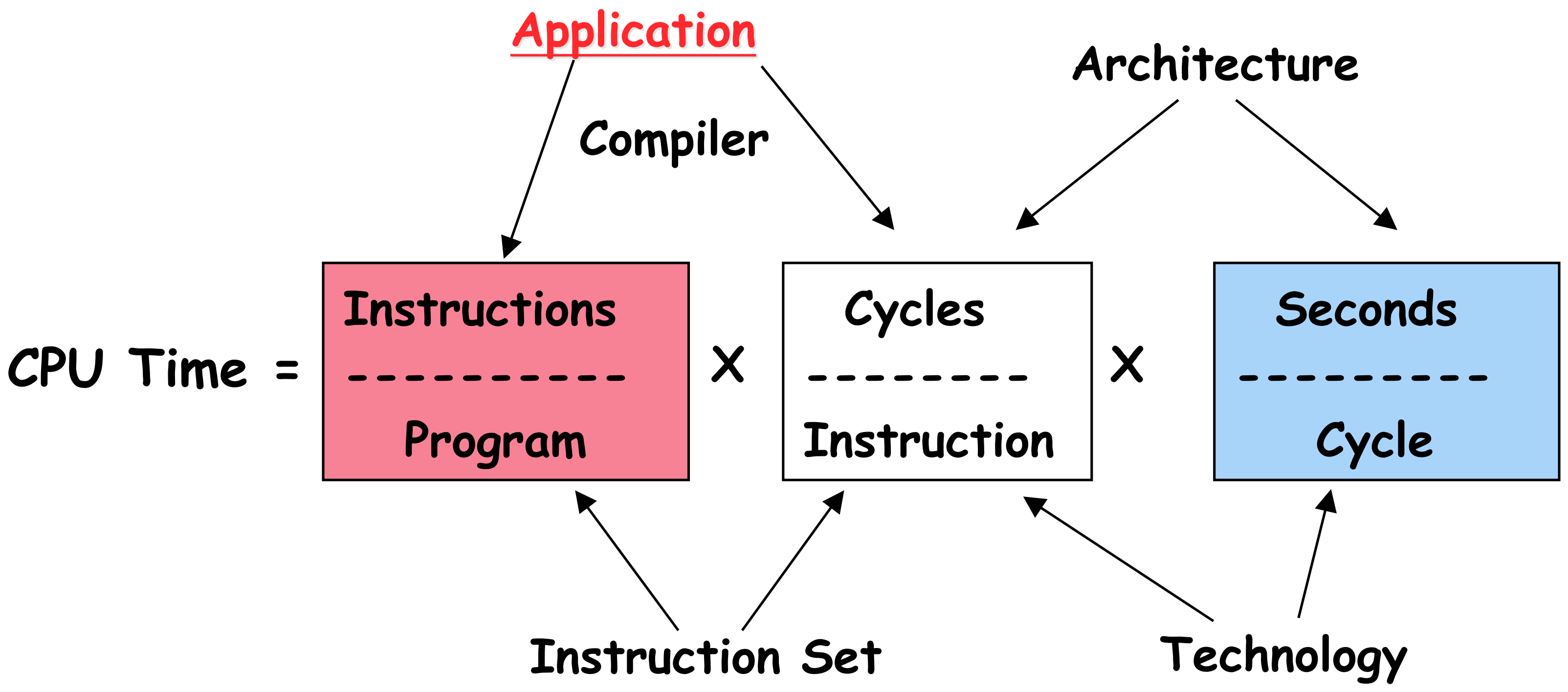
- Why?

Basic Single-CPU Performance Issues

- Instruction parallelism
  - Affected by CPU architecture, compiler, code
    - Architecture: Superscalar, pipelined processors - out of our control
    - Compiler: little improvement in ~10 years, out of our control
    - Code: significant limits to restructuring / optimization for real codes
- Memory speed
  - Affected by caches, technology, code
    - Caches: getting bigger, closer, but not enough for real codes
    - Technology: gap between CPU & memory speed is inevitable and constantly increasing (at alarming rates)
    - Code: significant limits to restructuring / optimization for real codes

Los Alamos
NATIONAL LABORATORY

"Fundamental Equation" of Serial Performance



$$\text{CPU Time} = N_{inst} * \text{CPI} * \text{Clock Period}$$

Los Alamos NATIONAL LABORATORY

Pitfalls

- Peak performance as an indicator of true performance.
  - (despite the fact that Q = ~4-5 X BM)
- Clock speed as an indicator of true performance.
- *Indirect* measures of performance as indicators of true performance (MIPS, MFLOPS, CPI, percentage of peak, cache hit ratio)
- Linpack as a measure of true performance.

Performance Analysis with Cycle Accounting

$$\text{CPU Time} = N_{inst} * CPI * \text{Clock Period}$$

$$= N_{inst} * \square\ CPI_i * \text{Clock Period}$$

- Useful *diagnostic* method: understand where the cycles are spent during execution

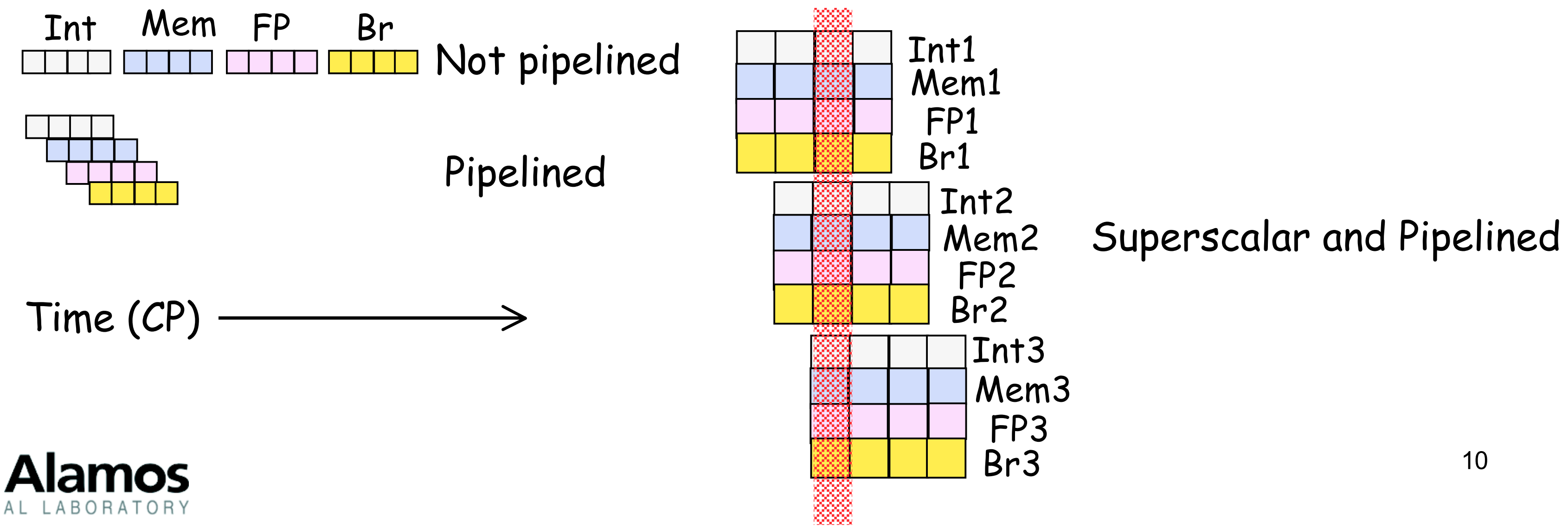
Important Example:

$$\text{CPU Time} = N_{inst} * \{CPI_{Compute} + CPI_{memory}\} * \text{Clock Period}$$

$$CPI_{stall}$$

ASCI Processors are Superscalar, Pipelined

- Superscalar: issue/execute >1 operation per clock period (CP) in separate functional units
  - Example: sgi MIPS R10000 (ASCI BlueMountain):
    - 1 integer,
    - 1 memory,
    - 1 Floating-Point multiply/add, and
    - 1 branch/conditional per CP.



Do the Codes Contain Optimal Instruction Mix?

- Answer: NO
- Examples:
  - Observed for PARTISN on ASCI BlueMountain:
    - ~ 3 memory references per FLOP
    - One memory reference every 2.4 instructions
  - Observed for SAGE on ASCI BlueMountain
    - 3 memory references per FLOP
    - One memory reference every 3 instructions

Los Alamos NATIONAL LABORATORY

Comparison with other "Benchmarks"

- Compare to Linpack: $O(n^3)$ FLOPS for $O(n^2)$ mem. ref's
- Compare to sPPM: 3.4 FLOPS for 1 mem. ref.

- Again, SAGE, Partisn: 1 FLOP for 3 mem. ref's.

- Conclusion: Code sequences not well matched to MIPS R10K, mostly due to memory ops
- Conclusion: "Benchmarks" not representative

Performance of BlueMtn. Using Ideal CPI

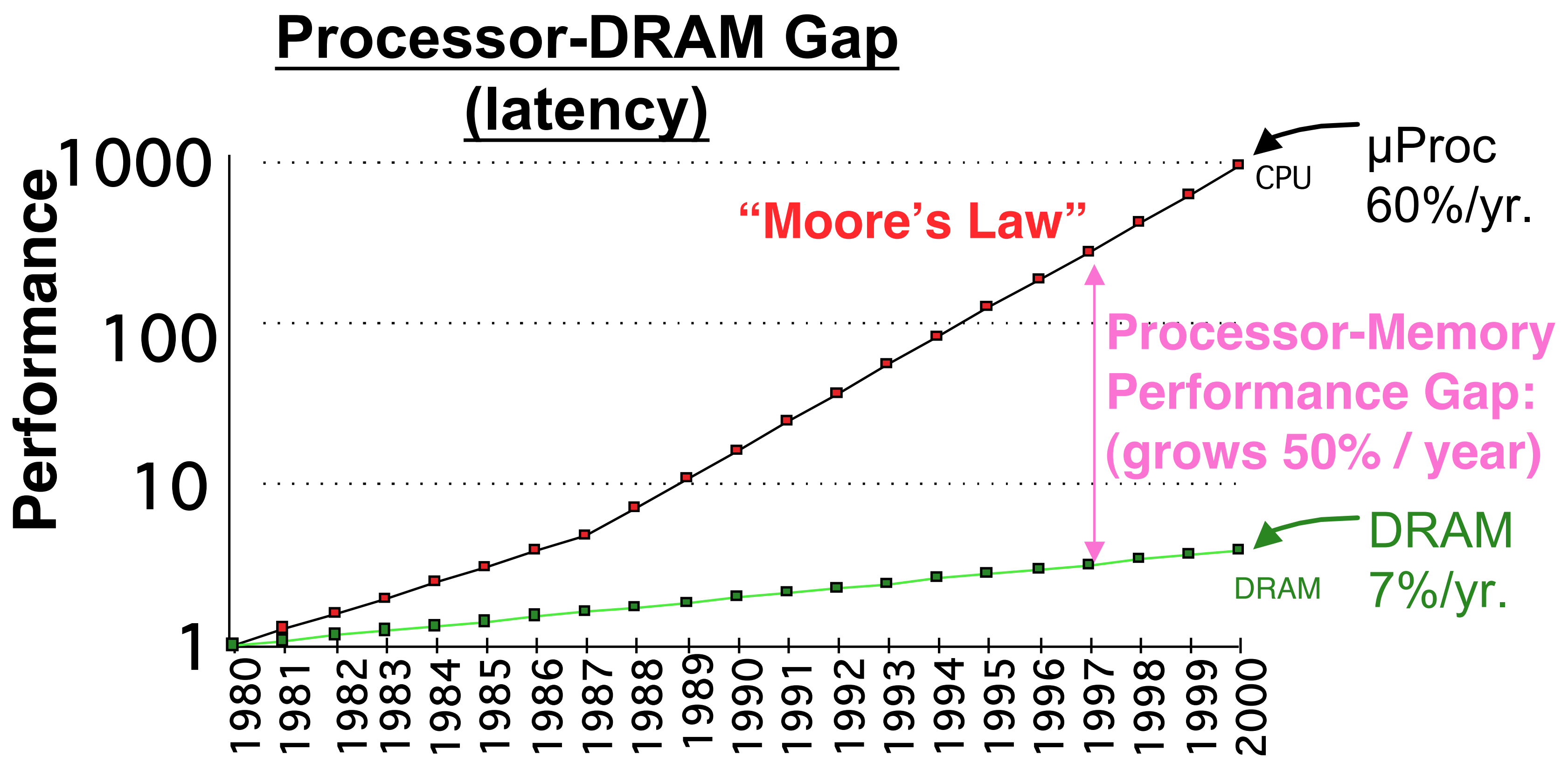
- **CPU Time = Ninst * {$CPI_{compute}$ + CPI_{stall}} * Clock Period**
- 4 instructions per clock period, therefore optimal CPI = 0.25 (smaller is better)
- Observed:

	$CPI_{compute}$
SWEEP	0.88
HYDRO	0.89
HYDRO-T	0.90
NEUT	0.77

This is CPIcompute, an estimate of CPI that eliminates memory latency effects.

We see that CPIcompute is still quite large compared with the ideal value.

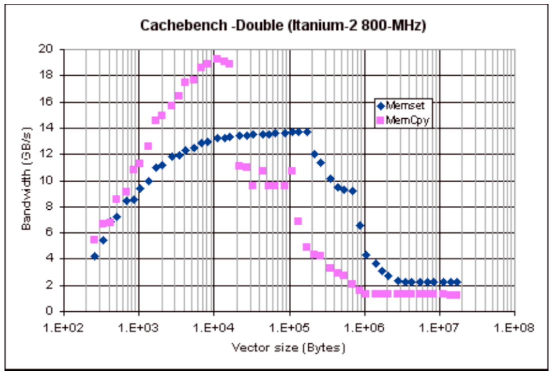
Where Else Does the Performance Go?



From D. Patterson, CS252, Spring 1998 ©UCB

Where Else Does the Performance Go?

- Memory performance is the key.
- Two important memory performance characteristics:
  - Latency: (Scalar) time between a processor's request for a datum and its delivery (from somewhere)
  - Bandwidth:



  - Which is more important? Depends on the code's locality.

Performance of BlueMtn. Using Full CPI

- CPU Time = Ninst * $\{CPI_{compute} + CPI_{stall}\}$ * Clock Period

- Observed:

	$CPI_{compute}$	Total CPI
SWEEP	0.88	1.6
HYDRO	0.89	1.3
HYDRO-T	0.90	0.95
NEUT	0.77	0.8

This table shows full, measured
CPI.

We see that in some cases, memory stall time comprises about 1/2 of CPI.

How Efficient are ASCI Processors?

- Examples:
  - Observed for PARTISN on ASCI BlueMountain:
    - 4191298240 FLOPS in 63.6 seconds = 65.9 MFLOPS (13% of peak)
  - Observed for SAGE on ASCI BlueMountain
    - 8679758816 FLOPS in 411.3459528 = 21 MFLOPS (4% of peak)
- Efficiences are low due to (low) memory speed and large numbers of memory operations. Large numbers of memory ops are inherent in codes with unstructured grids.
- Compare to Linpack (~70-80% efficiency)? Really, the comparison is meaningless.

Independant Data: SAGE Solver Rates

	BlueMtn. (Old)	BlueMtn. (New)*	Q (Old)	Q (New)
1-CPU	27.5 MFLOPS (5.5% of peak)	50 (10%)	117 (5.9%)	165 (8.2%)

- New: Code tuned by Rice U. to eliminate some gather/scatter ops.

See John Mellor-Crummey and John Garvin. Optimizing Sparse Matrix Vector Multiply using Unroll-and-jam *Proceedings of the Los Alamos Computer Science Institute Third Annual Symposium* October, 2002, Santa Fe, NM. Published on CD-ROM.

- NB: Solver consumes ~40-60% of a SAGE run typically.

Instruction Mix From Other Workloads

- SPECfp2000 Benchmark Suite, average instruction mix over 5 programs:
  - Memory: 39%
  - FP: 18%
  - Integer: 26%
  - Other: 17%

Hennessy & Patterson, "Computer Architecture," 3rd Edition.

- CPI for transaction-processing benchmark on a Q-like machine: 2.23 (optimal is 0.25)
- Conclusion: Superscalar architectures execute at low efficiencies on ASCI, on other scientific workloads, and on commercial workloads; problem is memory speed universally

Los Alamos
NATIONAL LABORATORY

Sweep3D: Initial Results From Itanium-2

$CPI = CPI_{Compute} + CPI_{stall}$

BM: 1.6 = 0.88 + 0.72

optimal = 0.25

$CPI_{compute}$ ~ 3.5X optimal

CPI_{stall} ~ 45% of CPI

It2: 0.63 = 0.23 + 0.40

optimal = 0.16

$CPI_{Compute}$ ~ 1.4X optimal

CPI_{stall} ~ 63% of CPI (85% of which is memory)

Does "Percent of Peak" Really Matter?

- SAGE (timing_b) on BM
  - (250 MHz, 500 MFLOPS Peak per CPU, 2 FLOPS per CP):
  - Time = 522 sec.
  - MFLOPS = 26.1 (5.2% of peak)

- SAGE (timing_b) on Itanium-2
  - (900 MHz, 3600 MFLOPS Peak per CPU, 4 FLOPS per CP):
  - Time = 91.1 sec
  - MFLOPS = 113.0 (3.1% of peak)

Los Alamos NATIONAL LABORATORY

Final Thoughts (1 of 2)

- Peak rate and clock rate say extremely little about actual performance.
- Per-processor efficiency is only an indirect measure of performance; we are only interested in TIME
- Be careful which benchmarks you use/regard.

Final Thoughts (2 of 2)

- 10 years of high-performance microprocessors:
  - Some improvement in compiler ability to transform complicated code sequences to enable instruction-level parallelism; little/no improvement for cache blocking
  - Data caches are growing but so are problem sizes
    - E.g., more levels of adaption desirable
  - ≤ ~10% of peak performance is the norm for a wide range of *real* codes
  - Expect this to continue in subsequent generations (IA-64, K8, etc.)
  - Code optimization could improve cache/processor utilization but algorithms constrain ultimate efficiency (sPPM vs. .e.g., SAGE)

ABSTRACT

- This talk features a discussion of performance-related issues for ASCI LANL codes. A two-part approach is used, separating the single- and multiple-processor issues, and the bulk of this talk concerns single-processor performance. Hardware counters are used to account for where the time is spent and both instruction level parallelism and memory-related stall time is quantified. The talk also contains ideas related to common pitfalls in measuring and reporting single-processor performance.

Los Alamos NATIONAL LABORATORY